Design Tools for Reconfigurable Hardware in Orbit (RHinO)

Matthew French[1], Paul Graham[2], Michael Wirthlin[3], Gregory Larchev[4], Peter Bellows[1], and Brian Schott[1]
mfrench@isi.edu, grahamp@lanl.gov, wirthlin@ee.byu.edu, glarchev@mail.arc.nasa.gov, pbellows@isi.edu, bschott@isi.edu
[1]University of Southern California, Information Sciences Institute, Arlington, VA
[2]Los Alamos National Laboratory, Los Alamos, NM
[3]Brigham Young University, Proto, UT
[4]NASA Ames Research Center, Moffett Field, CA

***Abstract*- The Reconfigurable Hardware in Orbit (RHinO) project is focused on creating a set of design tools that facilitate and automate design techniques for reconfigurable computing in space, using SRAM-based field-programmable-gate-array (FPGA) technology. These tools leverage an established FPGA design environment and focus primarily on space effects mitigation and power optimization. The project is creating software to automatically test and evaluate the single-event-upsets (SEUs) sensitivities of an FPGA design and insert mitigation techniques. Extensions into the tool suite will also allow evolvable algorithm techniques to reconfigure around single-event-latchup (SEL) events. In the power domain, tools are being created for dynamic power visualization and optimization. Thus, this technology seeks to enable the use of Reconfigurable Hardware in Orbit, via an integrated design tool-suite aiming to reduce risk, cost, and design time of multi-mission reconfigurable space processors using SRAM-based FPGAs.**

I. Introduction

SRAM-based FPGAs have become a tantalizing solution to processing on space-based payloads. They offer features that anti-fuse FPGAs do not, such as reprogrammability, embedded multipliers, and embedded processors, while also offering 5-10x more logic gates than anti-fuse based FPGAs. These features allow SRAM-based FPGAs to address resource multiplexing, fault tolerance, mission obsolescence and design flaws in on-orbit payloads that directly impact design cost and mission risk, while also providing better processing performance. The reprogrammability of SRAM-based FPGAs enables the use of multiple mission applications at different orbit stages, thus combining diverse mission attributes and payloads, reducing size, weight, and power. In-orbit faults in FPGAs can be corrected by updating the design to map out damaged resources. Computational and transmission resources can adapt dynamically to the environment based on established priorities. Furthermore, as mission-specific algorithms invariably improve over time, these improvements can be up-loaded. Design flaws in the system discovered after launch can be similarly corrected with firmware updates

However, a significant barrier to developing space-ready SRAM-based FPGA applications is the difficulty in designing for the rigorous constraints mandated by the operational environment. Two main issues limit the use of conventional FPGAs to such designs: (1) SRAM-based FPGAs are sensitive to radiation effects, namely, total dose (TD), latchup, and single-event-upsets (SEUs), because of their high proportion of memory structures; and (2) conventional FPGA designs are most often optimized for performance at the expense of disproportionately more power.

Current foundry process technology for Xilinx FPGA devices provides enough tolerance for a large number of ESE orbits for total dose and latching (no destructive latches have been reported), however the SEU presence is a major design/operational issue. The large amount of static memory within SRAM-based FPGAs, such as look-up tables, routing switch tables, etc., makes them sensitive to SEUs. Many techniques have been proposed and implemented for improving the reliability of digital circuits in the presence of SEUs. While traditional hardware redundancy techniques improve the reliability of FPGA designs (at the expense of increases in hardware, power, etc.), novel FPGA-specific techniques are required to address the unique vulnerabilities of SRAM-based FPGA architectures, while incurring less hardware overhead. Therefore, design automation tools evaluating and assessing the reliability of FPGA designs, inserting appropriate redundant hardware, and manipulating the low-level structures of the FPGA design are needed for robust operation and SEU and latch-up immunity.

Available FPGA synthesis tools optimize for speed or area, but not for real-time power consumption. Limited power estimation tools are available, such as Xilinx's Xpower; however, these are difficult to use and have limited utility to the actual FPGA design process. Accurate power estimates are only achievable after completing the entire iteration of the design cycle and provide no power optimization guidance. To make effective use of FPGAs in space, tools providing accurate power estimation and real-time optimization, operating on the FPGA's gate logic or on individual configurable logic blocks (CLBs), are needed; specifically: 1) to monitor power consumption early in the design process at a useful granularity (e.g., at CLB); 2) to aid in the design analysis that captures data-dependent transients as well as overall power consumption; and 3) to perform real-time constraint-driven automated power optimization similar to the way that area/timing constraints are accounted today.

Both the radiation-induced and power consumption effects are currently handled through manual intervention or, at best, through ad-hoc in-house tools. There is a real need in the community for validated design tool automation to raise the technology readiness level (TRL) of SRAM-based FGPA

user designs. The RHinO project is leveraging an established, open-source tool-suite that accepts output from commercially available synthesis tools to create tools that allow the developer of a space-based FPGA application to automatically analyze and optimize a Xilinx Virtex II FPGA circuit for space effects and power utilization. In the first year of this effort, tools were developed primarily for analyzing the circuits. The remaining years of this effort will focus more heavily on optimization techniques and validation. This paper will outline the JHDL tool suite and extensions made to it for the RHinO toolkit in section II. SEU Radiation effects are discussed in Section III, and power utilities are discussed in Section IV. Synergy with evolvable algorithms for mitigating SEL effects are discussed in Section V. Section VI will summarize the progress to date and draw conclusions.

II. The JHDL Tool Suite

A. Background

A critical technology to the RHinO tool suite is the open-sourced JHDL [1] FPGA design environment. This set of design tools allows a designer to create complex, high-speed FPGA circuit modules programmatically from within Java. A variety of design aids are available for circuits constructed in JHDL. The tool suite, shown in Figure 1, contains a digital circuit simulator, a circuit hierarchy browser, FPGA library primitives, and tools for exporting user designs into EDIF and VHDL. JHDL provides an open API into the circuit structure to facilitate the creation of application-specific design aids for viewing, revising, manipulating, or interacting with a user design. The integrated design aids, circuit API, and flexibility of JHDL make it an ideal tool for aiding the development of radiation-hardened and power-aware space-based FPGA designs. A variety of application-specific tools can be created to analyze and improve the reliability of FPGA circuits.

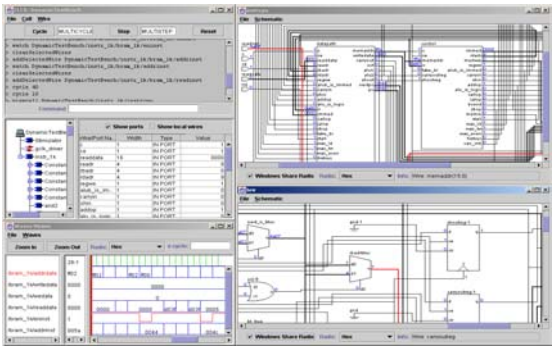

Figure 1. JHDL Tool Suite

Under this effort, RHinO is devising new features for JHDL, specific to space environments, that would enable SRAM-based FPGA payload developers to confidently manage the limiting on-board spacecraft design constraints for power, radiation effects, fault-tolerance, reliability, etc. A key goal of the effort is interoperation with existing commercial tool flows based on VHDL/Verilog, through seamless JHDL-EDIF translation. Alternatively, the user can work entirely in the JHDL design environment, using the RHinO power and SEU tools in concert with the normal JHDL features for simulation, netlisting, and runtime control, all within a single user interface.

B. RHinO Enhancements

The primary goal during the first year of this effort was to ensure that the JHDL infrastructure could support the desired SEU mitigation and power tool functionality. To accomplish this goal, efforts were divided into two major areas: JHDL tool modifications and the development of a robust EDIF netlist import infrastructure.

The JHDL tool suite is currently operational and fully supports the Xilinx Virtex2 FPGA. Although the tool is operational, there were a number of limitations within JHDL that needed revisiting in order to fully integrate the custom tools envisioned for this project efficiently. A new GUI API was developed to simplify the process of adding custom user interfaces to JHDL. This GUI API includes a new GUI event mechanism for custom applications to generate and catch JHDL related GUI events (i.e. mouse click, cell selection, etc.). This new API allows for example power visualization, schematic viewers, and waveform displays to share information and allow an FPGA developer unified time, signal, and power data across multiple viewpoints of their design.

The second major task during the first year was the development of a new EDIF netlist tool. Although JHDL provides support for EDIF parsing and netlisting, there were a number of disadvantages to the original implementation that limit its use in this project. First, the EDIF parser was not designed for memory efficiency and required far too much memory to parse full-chip designs. Second, the EDIF parser did not import multiple EDIF files for designs that import 3rd party IP. To address these issues, a new EDIF netlist tool was developed. This tool, developed independently from JHDL, supports the parsing of large EDIF netlists and can successfully merge multiple EDIF files. In addition, a JHDL export feature was added to allow this EDIF tool to generate JHDL designs and exploit all of the JHDL GUI and simulation tools. The EDIF netlist infrastructure is operational and several benchmark designs have been successfully parsed with this tool and converted into the JHDL environment, using EDIF files generated by Synplicity, CoreGen, System Generator, and ChipScope.

III. SEU Radiation Effects

A. Background

To further advance the TRL level of Virtex-II FPGAs for space applications, the RHinO project has a goal of improving the reliability of user designs in the presence of SEUs. SEUs are the main radiation concern since these FPGAs have been shown to have acceptable tolerance to TID as well as to SEL for low earth orbits (LEO). SEUs can occur in several memory structures on these SRAM-based FPGAs [2 3], namely in the support and control logic, the user design state, the programming memory (often called the *configuration memory*), and half-latches. Upsets in the

support and control logic can have a range of effects, from being fairly benign to totally erasing the contents of the FPGA configuration memory. Upsets in the user design state, such as in flip-flops and on-chip memories, may cause faults to occur in the user design's operation, while upsets in the configuration memory may change the user's design directly by changing the connectivity of logic or changing the logic functions themselves. Finally, Xilinx uses structures called half-latches to generate some constant logic values within user designs. These ubiquitous structures can be upset, but upsets in these structures are generally not detectable or easily correctable without taking the hardware off-line[4]. The RHinO toolkit concentrates on mitigating upsets in on-chip memories (flip-flops and on-chip RAMs), upsets in the configuration memory, and upsets in half-latches since they account for a majority of the SEUs experienced by these chips.

B. SEU Characterization

Because of the variety of upsets that may occur within these SRAM-based FPGAs, the first set of tools has focused on developing methods for better analyzing the complex effects of SEUs on designs. The main methods for doing this characterization have been to perform SEU emulation through hardware fault injection as well as using ionizing radiation produced by particle accelerators. SEU characterization performed by Xilinx and other partners using protons and heavy ions will be used as a baseline that will be extended upon[5, 6].

Since testing with ionizing radiation is challenging and expensive, the RHinO team is applying its experience with earlier SRAM-based FPGAs to develop an SEU emulation system that injects faults into user design's configuration file in software and then executes the corrupted configuration on hardware so a design's robustness can quickly be evaluated. The SEU emulation system, shown in Figure 2, provides the experimenter with complete control over the frequency and locations of faults within the configuration memory of the FPGA, which accounts for more than 90% of an FPGA's user observable state. Further, due to the increased speed at which experiments can be performed using SEU emulation, experimenters can do more extensive testing of designs in the presence of SEUs. As an example, SEU emulation systems developed for Virtex FPGAs allowed every bit of configuration memory to be upset *individually* within about 30 minutes. Upsets in the configuration logic can also induce upsets in half-latches used by designs, so SEU emulation can allow robustness testing of designs in the presence of half-latches SEUs as well. New capabilities of the Virtex-II architecture will also allow exploration of the possibility of introducing upsets into user design flip-flop state, something that was not feasible with previous device generations.

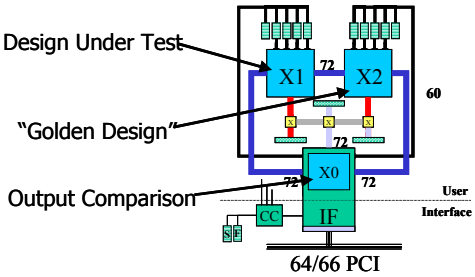

Figure 2. SEU Emulator Hardware Setup

C. SEU Mitigation

With the ability to characterize how designs fail through SEU emulation, methods for successfully mitigating against FPGA SEUs can be tested more rapidly. For mitigating against half-latch SEUs, a tool called *RadDRC II* was developed to analyze completed designs for the presence of these structures. Once discovered, the *RadDRC II* tool can remove the reliance on these structures by manipulating the user's design to use other constant-generation approaches, ones which are tolerant of radiation or those for which SEUs can be detected and corrected, as shown in Figure 3.

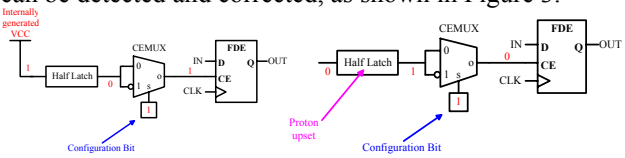

Figure 3. Half-latch Example

For SEUs in user state and configuration memory, development is beginning on tools for automatically introducing redundancy into designs, where appropriate, to increase design reliability. Techniques already exist for detecting and repairing SEUs in the configuration memory[7] and Xilinx provides a method and tool for applying triple-modular redundancy (TMR) to entire designs. RHinO is focusing on exploring ways of effectively trading off the costs of redundancy for reliability since some applications can withstand some level of faults and may not be implementable in full TMR due to the high area costs. An example of a lower cost mitigation approach is the mitigation of structures with feedback (state machines, counters, IIR filters, etc.) through TMR to prevent SEU-induced faults while allowing the feed-forward portions of the circuits to undergo upsets. The assumption is that faults in feedback structures are more critical since they may persist indefinitely and would require a system to be reset, while faults in the feed-forward structures will be flushed out of the circuits in a short period of time and do not dramatically affect the operation of the circuit, e.g., when the upsets appear as noise in sensor data. Further, the expectation is that for some designs the feedback structures in circuits will be less common than the feed-forward structures, thus, reducing the cost of the increased robustness to SEUs.

D. Current Status and Future Direction

To date, an SEU emulation system for Virtex-II has been built and is being tested. The framework for the SEU emulation system is being designed to be flexible enough to

easily include multiple SRAM FPGA families and multiple board architectures, as shown in Figure 4. The initial test system is built using commercially available FPGA boards from Xilinx (the AFX series of boards). Further, we have worked closely with Xilinx to understand the half-latch SEU sensitivities of Virtex-II FPGAs and have developed and are testing the *RadDRC II* tool for correctness and robustness.

During the next few months, the effectiveness of *RadDRC II* using proton radiation in addition to the SEU emulation system will be tested. Also during the next several months, a characterization study of SEUs for Virtex-II will be completed and work will begin on automatically identifying and mitigating SEU sensitivities in designs. Testing of the mitigation approaches and tools will be conducted with the SEU emulation system and proton radiation testing.

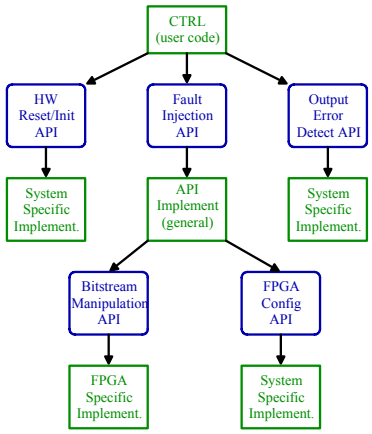

Figure 4. SEU Emulation Framework

IV. Power Optimization

A. Background

In space applications, power consumption is a vital specification, almost as important as clock speed performance. Current SRAM-based FPGA design tools have been created with only speed or area optimizations as their goal and only recently have accurate power measurement tools become commercially available. These tools, such as the Xilinx XPower tool, are limited in the content of the power information they provide, readability, and their entry point in the FPGA design flow. Pre-place and route estimates are currently performed by manually entering and estimating device utilization, toggling rates, and routing interconnect, a process highly dependant upon the designer's efficiency and expertise. Automated power measurements are available after going through a complete iteration of the entire design flow. However this process can be ad hoc and time consuming, as a designer needs to interact with multiple tools that generate multiple memory-hungry intermediary files. At this level, the machine-generated signal names are difficult to resolve with their functional level counterparts in order to make optimizations. Furthermore, no unified information is provided to the designer of a component's power utilization details, such as fan-out, component load, or interconnect. Power itself is reported as a single static value, making it difficult to isolate a data-dependant power spike, or different modes of operation. In other words, very little guidance is given to the designer on how to optimize if the power specifications were not met.

The RHinO tools aim to develop technology alleviating these limitations, by leveraging previous work characterizing the power consumption of Virtex-II devices, and developing tools for pre- and post-place and route dynamic power visualization. These tools will then be used to create tools for power optimization, and enhancing design entry tools, such as module generators, with power constraint specifications.

B. Dynamic Power Visualization

An extension of the JHDL tools for detailed power modeling, simulation and analysis of FPGA circuits was created. The RHinO power tool gives the application designer *per-net and per-cell power simulation from the beginning phases of design.* This strongly encourages the consideration of power dissipation as a primary design constraint.

The RHinO power tool exploits the extensible JHDL datatypes for representing the hierarchical structure and connectivity of circuits of an FPGA. The core datatypes are augmented with load (capacitance) estimates for each net and cell. ('Cell' refers to the atomic logic blocks of the FPGA – LUTs, registers, multipliers, RAM blocks, etc.). These capacitance estimates are calculated by a "power model" API. The power modeling infrastructure is pluggable, such that a variety of modeling algorithms can be used during the simulation. This allows the same simulator to work with the design at both the pre- and post-place and route design points, providing more accurate modeling with a fully routed design. Currently the tool utilizes a simple "unit load" model for un-routed designs which assumes that all nets and all cells have an identical load capacitance. This model has accuracy on par with the spreadsheet-based estimators distributed by Xilinx, but has the advantage of being fully automated and allowing dynamic power visualization. Development is ongoing on new power models for very detailed and accurate power estimates during functional simulation. These include heuristic modeling of net capacitance based on fanout, and importing post-place-and-route layout / capacitance information via the Xilinx XPower tool.

The power tool gives the designer two views to help visualize power dissipation issues and isolate the worst offenders. First, it presents a tree-view of the circuit and shows the cumulative power consumption of each module in the circuit, broken down hierarchically. The tree is sorted at every level by power consumption. This view allows the user to immediately pinpoint and focus optimization efforts on the parts of the circuit with the largest power impact. Even with simple power models (such as the "unit load" model), the designer can get a very good idea of the relative power consumption of the modules in the circuit.

The second tool view is a plot of the instantaneous power history of selected cells over time. The plot is continuously updated during the simulation. This view focuses on *power modalities*, helping the designer to focus on system-level

design issues to reduce the power consumption during various operating modes. For example an adaptive sensor device may cycle between 'active' (full-power) and 'sleep' (idle) modes based on the signal being monitored. If the active mode occurs irregularly, optimization of the lower-power modes may actually have a larger impact on total battery life. Conversely, some systems may have a limited current supply and may be interested in the maximum power used. The temporal view of power allows the user to identify and characterize these modalities in a visual way for any scenario, as seen in Figure 5.

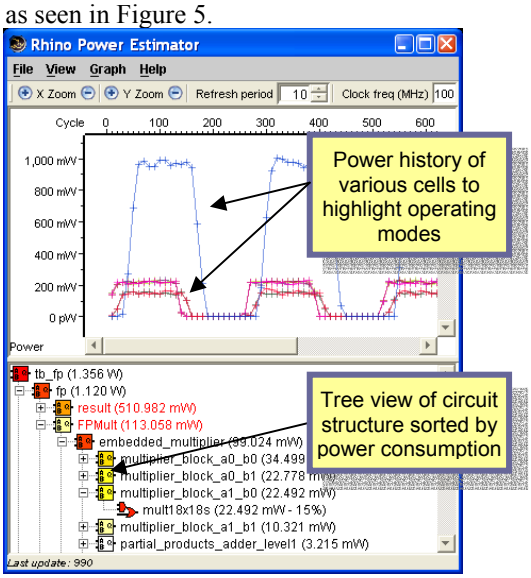

Figure 5. RHinO Power Estimator

C. Current Status and Future Direction

Final work is underway in developing accurate power models for functional simulation. Even the rudimentary model has pushed power measuring up to the forefront of the design flow, signaling potential trouble spots early on. Further work will also continue to push the power information into the JHDL schematic viewer as well.

With this effort completed development will turn from power analysis tools to power optimization tools. Most of the power in an FPGA is consumed by the low level routing resources[8] and the RHinO tools will address this via high level functional design and low level resource manipulation. The functional level tools seek to reduce power by re-ordering or shutting off components and their corresponding routing for brief periods. Here, the RHinO Power Estimator will be augmented to tag data flows to highlight unnecessary component toggling by analyzing clock enable, data flow, and multiplexer behavior in simulation. For example in many applications data may take only one of several possible data-dependant filter branches. Untaken branches should have their clock enable and clock lines disabled to reduce power, but finding all components and disable conditions can be difficult by hand.

Once functional level optimizations have been exhausted, lower level resource manipulation tools can make further refinements. These tools will analyze a simulation, prioritize high frequency, high load nets, and use relative placement macros (RPMs) to force the routing tools to utilize shorter interconnects for these routes. Further placement strategies could include clock tree distribution considerations.

Finally, the JHDL module generators will be expanded to include power (and SEU tolerance) as a design entry constraint, in addition to traditional throughput and precision constraints. The module generators can then use the known performance and capacitance information of FPGA components to create optimal modules for a particular design.

V. Evolvable Techniques

A. Background

Though SEUs are the primary radiation concern when dealing with SRAM-based FPGAs, for certain orbits and life-spans, TID and SEL do become a concern. Since it is virtually impossible to replace spacecraft components in-situ, there is a clear opportunity for fault-tolerant FPGA circuits.

Evolutionary algorithm (EA) methods hold promise in their ability to search across the space of FPGA configurations for those that can function in the presence of certain types of faults. Since SRAM-based FPGAs are fully reprogrammable, it is possible to restore the functionality of the compromised FPGA by re-routing a circuit around corrupted resources, a property which the RHinO team is exploring.

The evolvable methods under research tackle the problem of Single Event Latchups (SELs.) Both sequential and combinational circuits have been successfully evolved on a physical FPGA[9,10,11]. The sequential circuit was a Quadrature Decoder, 4-state state machine. The combinational circuit evolved was a 3-bit multiplier. Both circuits have been evolved from scratch, which can be thought of as the most extreme case of fault repair. However, in case of Quadrature Decoder, the repair of a previously working circuit in the presence of faults has also been demonstrated (permanent stuck-at faults were simulated.) The genetic algorithm under investigation evolves both circuit logic and routing (as opposed to just logic), which is important since on a typical FPGA a majority of transistors are dedicated to interconnect. The evolved multiplier is also the first 3-bit multiplier to be evolved on a physical FPGA (all the previous ones were evolved in simulation.) This innovation is important because evolving a circuit on a physical FPGA automatically takes into account all the physical features (such as faults) of that FPGA; thus, such evolution is more relevant to fault tolerance than evolution in simulation.

B. Current Status and Future Directions

The evolutionary algorithm is being further refined to handle larger, more sophisticated circuits using the Virtex-II FPGA architecture, with the end-goal of showing that the algorithm is robust enough to solve SELs in the project's benchmark 3x3 image convolution kernel and other real-life

applications. When a fault occurs in a large, complex circuit, the plan is to isolate the fault to a simpler component and then to re-evolve the component. Operating on full-sized applications on FPGA hardware will be a major thrust of this effort.

All of the team's FPGA evolution work to date uses bitstring chromosomal representation. This representation is the simplest but also the least efficient one. As larger circuits are considered, the shortcomings of bitstring representation will become more apparent. Therefore, work is underway to change the algorithm to generative (tree-like) representation. By being more conducive to component reuse, generative representation shortens the chromosome length and makes the evolution of the larger circuits more manageable.

In the upcoming year, the algorithm will be proven on larger application circuits. Further on, this algorithm will be integrated with the rest of the RHinO toolkit. Ultimately, the evolutionary algorithm will evolve circuits which are not only immune to existing SELs but also use guidance from the rest of the RHinO toolkit to create circuits that are SEU tolerant as well.

VI. Conclusions

In the first year of this effort, considerable effort was spent developing the baseline tool infrastructure, radiation analysis tools and power analysis tools. The JHDL infrastructure was enhanced to allow better GUI development and fully support EDIF file import from a variety of commercial synthesis tools. In the radiation arena, the SEU emulator was developed to allow rapid analysis of SEU effects on a design and facilitate development of SEU mitigation schemes. This tool then allowed full testing of half-latch mitigation techniques on Virtex-II and the development of a half-latch mitigation tool. The results of these studies are being shared with evolvable techniques for SEL mitigation in the form of cost tables, such that newly evolved circuits do not contain SEU sensitivities. Finally, a power API was developed, allowing the development of a singe power visualization tool capable of using different power models for pre- and post-routed designs. The RHinO team plans to have a beta release of these tools available in the next month. The next year's efforts seek to leverage the infrastructure, radiation analysis tools, and power analysis tools, to focus on radiation mitigation and power optimization tools.

References

[1] "Adaptive Computing Systems"; 1997- 2003 DARPA effort; see www.jhdl.org

[2] Carl Carmichael, Earl Fuller, Phil Blain, and Michael Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications", Proceeding of the Military and Aerospace Programmable Logic Devices International Conference (MAPLD), Sept. 1999, Laurel, MD, pp. C2.1-8.

[3] Michael Caffrey, Paul Graham, Michael Wirthlin, Eric Johnson, and Nathan Rollins, "Single-Event Upsets in SRAM FPGAs", Proceedings of the 5th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD), Sept. 2002, pp. P8.1-6.

[4] Paul Graham, Michael Caffrey, Michael Wirthlin, D. Eric Johnson, and Nathaniel Rollins, "SEU Mitigation for Half-Latches in Xilinx Virtex FPGAs", *IEEE Transactions on Nuclear Science*, Dec. 2003, 50(6), pp. 2139-2146.

[5] Candice Yui, Gary Swift, and Carl Carmichael, "Single Event Upset Susceptibility Testing of the Xilinx Virtex-II FPGA", Proceedings of the 5th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD), Sept. 2002, pp. P29.1-6.

[6] CC. Yui, G.M. Swift, C. Carmichael, R. Koga, and J.S. George, "SEU Mitigation Testing of Xilinx Virtex-IIFPGAs", 2003 IEEE Radiation Effects Data Workshop, July 21-25, 2003, Monterey, CA, pp. 92-97.

[7] Carl Carmichael, Michael Caffrey, and Anthony Salazar, "Correcting Single-Event Upsets through Virtex Partial Configuration", XAPP216 (v1.0), June 2000, Xilinx Corporation.

[8] L. Shang, A. Kaviani, K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family," FPGA '02, Monterey, CA, February 2002.

[9] J.Lohn, G.Larchev, R.DeMara, "A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs," in Proceedings of Evolvable Systems: From Biology to Hardware, ICES 2003, March 2003, Trondheim, Norway.

[10] A.Thompson, "Silicon Evolution", in Proceedings of Genetic Programming Conference 1996, July 28-31, 1996, Stanford, CA.

[11] V.K.Vassilev, D.Job, J.F.Miller, "Towards the Automatic Design of More Efficient Digital Circuits," in Proceedings of The 2nd NASA/DOD Workshop on Evolvable Hardware, July 13-15, 2000, Palo Alto, CA.